Matching in Regular Bipartite Graphs

- For any positive integers n and d, let $\mathcal{G}_{n,d}$ denote the set of all d-regular bipartite graphs (U, V, E) where $|U| = |V| = n$ and $E \subseteq U \times V$
- Using Hall's theorem, it is easy to argue that any bipartite graph in $\mathcal{G}_{n,d}$ admits a perfect matching
  - Indeed, the edge set can be partitioned into d perfect matchings
- We will present a fast randomized algorithm for computing a perfect matching in a given bipartite graph in $\mathcal{G}_{n,d}$

High-Level Description of the Algorithm

- Recall the elementary flow-based approach to computing a maximum cardinality matching of an arbitrary bipartite graph
  - Repeatedly augment a 0-1 flow by one unit
  - A 0-1 flow of value ℓ corresponds to a matching of cardinality ℓ
  - For a d-regular bipartite graph, the running time is $O(dn)$ per iteration for an overall running time of $O(dn^2)$
- Our plan is to use a random walk to find an augmenting path
  - If k vertices on each side remain unmatched, we will show that the expected time for the random walk to find an augmenting path is $O(n/k)$
  - This results in an overall expected running time of $O(n \log n)$

Analysis of a Single Augmentation: Basic Definitions

- Assume that our bipartite graph $G = (U, V, E)$ belongs to $\mathcal{G}_{n,d}$
- Let U_0 (resp., U_1) denote the unmatched (resp., matched) vertices in U
- Let V_0 (resp., V_1) denote the unmatched (resp., matched) vertices in V
- Let μ denote the current matching
  - For any vertex u in U_1, we write $\mu(u)$ to denote the match of u in V_1
  - For any vertex v in V_1, we write $\mu(v)$ to denote the match of v in U_1

The Random Walk

- Instead of introducing a source and sink, we will start at a uniformly random node in U_0, and we will terminate once we reach any node in V_0
  - From a node u in U_0, we go to a uniformly random node in $\Gamma(u)$
  - From a node u in U_1, we go to a uniformly random node in $\Gamma(u) - \mu(u)$
  - From a node v in V_1, we go to $\mu(v)$
- We can easily prune the random walk walk to obtain a simple alternating path for augmenting the matching μ
- For any node x, let $b(x)$ denote the expected number of "back" edges (i.e., edges from V to U) traversed before the walk terminates, assuming that we are currently at node x

Analysis

- Lemma 1: For any v in V, we have

$$b(v) = \begin{cases} 0 & \text{if } v \in V_0 \\ 1 + b(\mu(v)) & \text{if } v \in V_1 \end{cases}$$

  - If v belongs to V_0 then the walk terminates
  - If v belongs to V_1 then we traverse the back edge from v to $\mu(v)$, and the walk continues from $\mu(v)$

Analysis (cont'd)

- Lemma 2: For any u in U_0, we have

$$d \cdot b(u) = \sum_{v \in \Gamma(u)} b(v)$$

  - From a vertex u in U_0, we select a uniformly random vertex v in $\Gamma(u)$, traverse the "forward" edge from u to v, and then continue the walk from v
  - Hence $b(u)$ is equal to $\frac{1}{d} \sum_{v \in \Gamma(u)} b(v)$

Analysis (cont'd)

- Lemma 3: For any u in U_1, we have

$$d \cdot b(u) = -1 + \sum_{v \in \Gamma(u)} b(v)$$

  - From a vertex u in U_1, we select a uniformly random vertex v in $\Gamma(u) - \mu(u)$, traverse the "forward" edge from u to v, and then continue the walk from $\mu(v)$
  - Hence $b(u)$ is equal to $[-b(\mu(u)) + \sum_{v \in \Gamma(u)} b(v)]/(d-1)$
  - Equivalently, $(d-1)b(u) = -b(\mu(u)) + \sum_{v \in \Gamma(u)} b(v)$
  - The claim follows since $b(\mu(u)) = 1 + b(u)$

Analysis (cont'd)

- Lemma 4: $d\sum_{u\in U} b(u) = (d-1)|\mu| + d\sum_{u\in U_1} b(u)$
  - By Lemmas 2 and 3, the LHS is equal to $-|\mu| + \sum_{u\in U}\sum_{v\in\Gamma(u)} b(v)$
  - Since the graph is d-regular, this is equal to $-|\mu| + d\sum_{v\in V} b(v)$
  - By Lemma 1, this is equal to $-|\mu| + d(|\mu| + \sum_{u\in U_1} b(u))$
  - The claim follows

Analysis (cont'd)

- Let T_k denote the expected length of the random walk when $|U_0| = |V_0| = k$ and hence $|\mu| = n - k$
- Thus $T_k = 1 + \frac{2}{k} \sum_{u \in U_0} b(u)$
- Lemma 4 implies that $d \sum_{u \in U_0} b(u) = (d-1)(n-k)$
- Hence $T_k = 1 + \frac{2(d-1)(n-k)}{dk} < 1 + 2n/k$
- Thus the overall expected running time is

$$O\left(\sum_{1 \leq k \leq n} T_k\right) = O(nH_n) = O(n \log n)$$

How Robust is this Result?

- Intuitively, one might expect the $O(n \log n)$ expected time bound to hold for graphs that are "nearly" regular (and contain a perfect matching, say)
- Somewhat surprisingly, even a small deviation from regularity can dramatically worsen the expected running time
- We will demonstrate a bad example where we add a single additional edge to a 3-regular bipartite graph
- We begin by discussing a random walk on a cycle

A Random Walk on a Cycle

- Consider a cycle with n nodes indexed from 0 to $n-1$ in clockwise order
- Consider a random walk starting at node i
  - At each step of the walk, we use an independent fair coin flip to determine whether to move one step clockwise or one step counterclockwise
  - The walk ends when we reach node 0
  - It is not hard to prove that the expected number of steps is $i(n-i)$
  - Thus if we start at $\Theta(n)$ distance from node 0, the expected running time is $\Theta(n^2)$

A Random Walk on a Cycle (cont'd)

- Suppose that we start the random walk at a node $i \neq 0$
- Let E denote the event that the random walk terminates by moving from node 1 to node 0
- Thus $\neg E$ is the event that the random walk terminates by moving from node $n-1$ to node 0
- It is not hard to prove that $\Pr(E) = (n-i)/n$

A Bad Example

- Let n be an even positive integer, and consider the following bipartite graph $G = (U, V, E)$ in $\mathcal{G}_{n,3}$
  - Index the nodes in U (resp, V) from 0 to $n - 1$
  - Connect each node i in U to nodes $(i - 1) \bmod n$, i, and $(i + 1) \bmod n$ in V
- Construct $G' = (U, V, E')$ from G by setting E' to $E + (u, v)$ where u is the node with index 0 in U and v is node the node with index $n/2$ in V
- Assume that our current matching μ matches node i in U to node i in V for $1 \leq i \leq n$
- What is the expected running time of the random walk in G' that starts at the node with index 0 in U and terminates at the node with index 0 in V?

A Bad Example (cont'd)

- Starting at the node with index 0 in U, there is a $1/4$ chance that we move to the node with index $n/2$ in V in the first step
- If we do this, then based on our discussion of the random walk on a cycle, the expected number of steps to reach the node with index 0 in V is $\Omega(n^2)$

A "High Probability" Result?

- Let us define "with high probability" to mean "with arbitrary inverse polynomial failure probability"
- Does the random walk algorithm have $O(n \log n)$ running time with high probability?
- Consider the 3-regular graph G associated with the previous construction
- Based on our discussion of the random walk on a cycle, there is an $\Omega(1/n)$ chance that the random walk on G will reach the node with index $n/2$ in V
- Thus the augmentation takes $\Theta(n^2)$ time with probability $\Omega(1/n)$

An "Abort-and-Restart" Variant

- A natural variant of the random walk algorithm does achieve $O(n \log n)$ time with high probability
  - Suppose $|U_0| = |V_0| = k$
  - Recall that the expected length of the random walk is less than $1 + 2n/k$
  - Markov's inequality implies that the random walk terminates within $2(1 + 2n/k)$ steps with probability at least $1/2$
  - If the random walk fails to terminate within $2(1 + 2n/k)$ steps, we restart it from the beginning

Analysis of the Abort-and-Restart Variant

- We claim that the overall running time of the abort-and-restart variant of the random walk algorithm is $O(n \log n)$ with high probability
- To establish this, we instead analyze a related game
  - It is straightforward to verify that the bound we establish for the game implies the claim

The Game

- A video game has $\log_2 n$ "levels" indexed from 1 to $\log_2 n$, where n is a power of 2
- You start play at level 1 with "energy" $cn \log_2 n$ and repeatedly fight against one monster at a time
- Each time you fight with a monster on any level, you have a fifty percent chance of defeating the monster
- To advance from level ℓ to level $\ell + 1$, you need to defeat $n2^{-\ell}$ level-ℓ monsters
- Whenever you fight with a level-ℓ monster, your energy is reduced by 2^{ℓ}, whether or not you defeat the monster
- You win if you complete all of the levels without running out of energy

Analysis of the Game

- The expected number of level-ℓ monsters faced is $2n2^{-\ell}$
- By Markov's inequality, the probability that you need to fight more than $4n2^{-\ell}$ level-ℓ monsters is at most $\frac{1}{2}$
- The cost of facing $4n2^{-\ell}$ level-ℓ monsters is $4n$
- Let the random variable Z denote the number of independent flips of a fair coin required to get $\log_2 n$ heads
- Observe that the probability you lose the game is upper bounded by the probability that $4nZ \geq cn \log_2 n$, i.e., that $Z \geq (c/4) \log_2 n$

Analysis of the Game (cont'd)

- The probability that Z exceeds $(c/4)\log_2 n$ is equal to the probability that we get fewer than $\log_2 n$ heads in $(c/4)\log_2 n$ independent flips of a fair coin
- Recall that for a random variable X drawn from $B(k, 1/2)$ and any δ in $[0, 1]$, we have

$$\Pr(X \leq (1-\delta)k/2) \leq \exp(-\delta^2 k/2)$$

- Using the above bound, the probability Z exceeds $(c/4)\log_2 n$ is upper bounded by an inverse polynomial in n where the exponent is quadratic in c
- Thus the probability you lose the game is upper bounded by an arbitrary inverse polynomial for a sufficiently large choice of the constant c